[0262] The meta-implementation layer defines one implementation interface for each descriptor interface. As long as each implementation accesses the other implementations through the meta-implementation layer, different implementations can be mixed together even if they are on different platforms, written in different languages, and using different implementation paradigms (like relational database structures versus object-oriented structures versus structured programming).

[0263] An implementation actually performs functionality and creates new structures for storing and retrieving data. The implementations of the present invention are interfaces that an implementation must implement to participate in the meta-implementation layer. These interfaces are the same interfaces an accessor must implement. A virtual implementation differs from an accessor by being the implementation rather than delegating to an implementation. Different virtual implementations may implement the same implementation interface differently.

[0264] An implementation has a one-to-one association relationship with a descriptor that is the description for which a given implementation is an implementation. An implementation has a zero-to-one association relationship with a description that is a description of the implementation and use. An implementation may include the following operation: newInstance(ParameterList) that creates a new instance of this implementation. An implementation change event is fired when the implementation is changed in any way.

[0265] Implementations define the rules something must follow to be an instance of that implementation. An instance holds a reference to the rules it follows. An instance cannot change the implementation to which it points. The implementation is set at construction.

[0266] An instance has a one-to-one relationship with an implementation that is the implementation defining the instance.

[0267] A failure descriptor in the metamodel layer describes the failures an operation throws, not the failure itself. The failure itself is implemented using a model implementation. To access a failure, use the model implementation for the failure.

[0268] A constraint implementation of the present invention is an implementation holding a model implementation used to perform a specific restriction. The model must implement the constraint interface to participate as a constraint.

[0269] A constraint implementation has a one-to-one association relationship with a model implementation that is the identity of the model used to implement the particular constraint associated with the constraint implementation.

[0270] The configuration in a constraint descriptor gives the identity of the model implementation for a given constraint. That identity is used to retrieve the model implementing the constraint from the metamodel repository. No additional relationships are added by the constraint implementation.

[0271] A constraint implementation may include the following operations: getModelIdentity( ) that gets the identity of the model this constraint uses as a real implementation, and getModelImplementation( ) that gets the model that implements the constraint. That model will be used for constructing new instances of the constraint. The model must implement I_ConstraintInstance.

[0272] A constraint implementation defines no additional signals.

[0273] Composite constraints exist to provide AND/OR operations to combine two or more simple constraints into a more complex constraint. Since each constraint is evaluating a Boolean expression, composite constraints simple use logic to combine these expressions.

[0274] The AndConstraintImplementation contains one data implementation for the constraints which will be combined together using the AND Boolean operation.

[0275] An access constraint implementation of the present invention is a feature implementation and a constraint implementation. Access constraints check the current thread of control to see if it contains an authorized user. If no user has authorized, the constraint may perform a callback requesting the user authenticate. Otherwise the constraint fails, the operation stops, and a failure is thrown. If a user has been authenticated, an access constraint looks to see if the user has the appropriate credentials required to execute the current operation. If the user does not hold these credentials, the operation stops and throws a failure. An access constraint defines a single data implementation holding credential information that will be checked by the access constraint. Some access constraints will create the credentials based on the parent of which the access constraint a feature. Others will have a static set of credentials.

[0276] An occurrence constraint implementation of the present invention extends a feature implementation and constraint implementation. Occurrence constraints examine the number of items allowed in a specific location. A MinimumOccurrenceConstraint would require that there exist at least x instances, where x is a value set in the constraint descriptor. A MaximumOccurrenceConstraint would require that no more than x instances exist, where x is a value set in the constraint descriptor. Many other occurrence constraints are possible.

[0277] An occurrence constraint implementation is a simple extension of constraint implementation and feature. Below occurrence constraint is a sub-implementation of occurrence constraint that restricts the number of occurrences to between a minimum value and a maximum value. The range constraint defined two data implementations maximum value and minimum value. The range constraint expects two data instances to be set on the constraint instance. The range constraint implementation may further enforce the restriction that the maximum data instance is always greater than or equal to minimum data instance by creating a premutator operation on the minimum value and maximum value data implementations.

[0278] A value constraint implementation of the present invention extends feature implementation holding and constraint implementation. Value constraints inspect a data object and compare a value to a set of requirements. Value constraints can limit numbers to certain ranges. For example, a RangeValueConstraint can ensure a percentage parameter is always between 0 and 100. A PasswordConstraint could ensure a password have at least 8 characters